Real-time MTL with durations as SMT with applications to schedulability analysis

André de Matos*, Martin Leucker*, David Pereira†, and Jorge Sousa Pinto‡
* ISP, University of Lübeck, Germany † CISTER/INESC TEC, ISEP, Polytechnic Institute of Porto, Portugal
‡ HASLab/INESC TEC & Universidade do Minho, Portugal

***Abstract*—This paper introduces a synthesis procedure for the satisfiability problem of RMTL-$\int$ formulas as SAT solving modulo theories. RMTL-$\int$ is a real-time version of *metric temporal logic* (MTL) extended by a duration quantifier allowing to measure time durations. For any given formula, a SAT instance modulo the theory of arrays, uninterpreted functions with equality and non-linear real-arithmetic is synthesized and may then be further investigated using appropriate SMT solvers. We show the benefits of using RMTL-$\int$ with the given SMT encoding on a diversified set of examples that include in particular its application in the area of *schedulability analysis*. Therefore, we introduce a simple language for formalizing schedulability problems and show how to formulate timing constraints as RMTL-$\int$ formulas. Our practical evaluation based on our synthesis and Z3 as back-end SMT solver also shows the feasibility of the overall approach.**

***Keywords*-metric temporal logic; schedulability analysis; constraint programming; satisfiability modulo theories;**

I. Introduction and Motivation

Temporal logics are the standard tool for specifications and have proven to be useful in many different application areas. The best-known logic is Pnueli's Linear-time Temporal Logic (LTL) [28]. However, LTL has two major deficits limiting its application for certain aspects: First, LTL comes with a qualitative notion of time as it talks only about steps of the system. As consequence, LTL is not adequate for real-time specifications whenever the system's runs have to be modeled with *timed interval sequences* or as *flows* with the domain in $\mathbb{R}_{\geq 0}$. Second, LTL is qualitative in the sense that it does not allow counting of how often or for how long a property was true within a trace. In this paper, we study *restricted metric temporal logic with durations* (RMTL-$\int$) [9] which is interpreted over real-time traces and allows measuring time durations.

A widely known extension of LTL for potentially dealing with real-time is *metric temporal logic* (MTL) in which the modalities of LTL are augmented with timing constraints [1]. MTL formulas can be interpreted over a variety of temporal models such as discrete (e.g., $\mathbb{N}$, $\mathbb{Z}$) [13], [2] and dense (e.g., $\mathbb{R}$) [21], [6], [33], [18] time domains. Metric operators defined over discrete time can be regarded as simple *syntactic sugar*, since they are a succinct way of expressing metric constraints that can be encoded using the LTL's *next* modality.

Dense-time MTL operators are commonly classified in terms of *pointwise* and *continuous* semantics. The pointwise semantics is evaluated along possibly infinite sequences of timed words, i.e., sequences of pairs $(e_0, t_0)(e_1, t_1)\ldots$, where e_i are events/propositions belonging to a non-empty alphabet Σ and $t_i \in \mathbb{R}_{\geq 0}$ are the time instants of the events' occurrence. The continuous semantics is evaluated over possibly infinite signals: Given a set of propositions P, a signal is a function $f : \mathbb{R}_{\geq 0} \to 2^P$ mapping $t \in \mathbb{R}_{\geq 0}$ to the set $f(t)$ of propositions holding at time t. A restriction of the continuous semantics for evaluating timed interval sequences is also known as an *interval-based semantics*, or in other words, a *continuous semantics* with finite variability. Timed interval sequences are sequences of pairs $(e_0, l_0)(e_1, l_1)\ldots$, where l_i are contiguous, non-overlapping intervals with real or rational bounds, forming a sequence of intervals in $\mathbb{R}_{\geq 0}$. In this paper, we aim for interval-based semantics with finite variability.

Our logic RMTL-$\int$ [9] is appropriate for reasoning about hard real-time systems, since it extends the expressiveness of MTL with a fragment of classical logic, the *first order logic of real numbers* (FOL$_{\mathbb{R}}$)[23], which allows us to count time.

While [9] concentrated on the use of the logic in terms of runtime verification, this paper studies its (practical) satisfiability problem by means of SAT solving modulo theories (SMT) using corresponding SMT solvers. Modern SMT solvers have been the target of immense effort in the last years. Even though the development of techniques and tools for nonlinear reasoning have not received too much attention during this period, a notable approach using Tarski queries has been described by Jovanović and Moura [23], [11]. This approach introduces a clear advantage over SMT solvers by including support for the decidable theory of the *real closed fields*. Other efforts for efficient reasoning about polynomials can be found in Yices 2 [12] and Alt-Ergo [30]. From the computer algebra standpoint, Mathematica and Maple have been the tools of choice for symbolic reasoning in nonlinear arithmetic. Although SMT solvers seem particularly suited to be used in classic problems in the area of real-time systems, in practice this has not yet happened a lot. This work will hopefully contribute towards their adoption by this community.

More specifically, we present a translation schema that allows to synthesize an SMT instance for a given RMTL-$\int$ formula that is equi-satisfiable. For this, we use the theory of arrays, uninterpreted functions with equality and non-linear real-arithmetic. Any model found by the SMT solver can be translated back into an interval trace satisfying the original formula. We show the benefits of using RMTL-$\int$ with the given SMT encoding on a diversified set of examples that include in particular its application in the area of *schedulability analysis*.

978-1-7281-4086-5/20/$31.00 ©2020 IEEE
DOI 10.1109/TASE49443.2020.00016

For almost forty years, a diversity of schedulability tests for hard real-time systems have been proposed to address the constraints imposed by the timing predictability required by this class of systems to satisfy safety and dependability requirements. These tests vary considerably in their complexity, expressivity, and target scheduling policies (e.g., fixed priority, earliest deadline first, round-robin, or even work stealing). The literature [3], [17] on hard real-time schedulability analysis reveals that generally, schedulability testing works by assuming a worst-case scenario and checking that each of the involved tasks gets a sufficient allocation of shared resources so that the corresponding job instances always complete before their deadlines. Formal verification approaches for schedulability analysis have been considered along these years, including the use of temporal logics [36], [32]. Temporal logics have been used as a formalism for specifying qualitative ordering constraints on the observable traces.

We show that RMTL-$\int$ is particularly suited as a building block for schedulability analysis. We introduce a simple language for formalizing schedulability problems and show how to formulate timing constraints as RMTL-$\int$ formulas. For the latter, we developed a tool translating the specification into an SMT problem for the SMT solver Z3 [10]. In this way, a schedule for the original problem can be found, if it exists. Our practical evaluation shows the feasibility of the overall approach.

Contributions of the paper: In this paper, we introduce a new sound synthesis mechanism that transforms RMTL-$\int$ formulas into the input language of modern SMT solvers; we propose a formalization of periodic resource models, which allows us to provide a push button approach for checking, in practice, schedulability of embedded real-time systems; and a synthesis tool[1].

Structure of the paper: The paper is organized as follows: Section II recalls definitions of both the syntax and semantics of RMTL-$\int$ as well as the lambda calculus used to support the translation. Sections III and IV describe, respectively, the synthesis mechanism and the formalization of the schedulability analysis of resource models (a widely used technique to ensure time isolation). Section V presents the evaluation of the synthesis mechanism, first for a chosen set of formulas, and later for the schedulability analysis of resource models. Finally, Section VI addresses related work and Section VII draws some conclusions and discusses future work.

II. Preliminaries

Before introducing the synthesis of RMTL-$\int$ formulae, we present an overview of the syntax and semantics of this logic and also the translation language used during the synthesis process. Our approach first translates RMTL-$\int$ into λ-expressions, which are then encoded into SMT solvers in the form of SMT-LIBv2 specifications. The choice to take λ-calculus as an intermediate language was due to its ability to describe the interpreted functions and also to convert them into uninterpreted ones using a simple translation pattern. In this way, the λ-calculus can be seen as an elegant computation model to synthesize temporal logic for different purposes such as runtime verification and/or shallow embedding in (automated) theorem provers.

[1]Available at https://github.com/anmaped/rmtld3synth

A. RMTL-$\int$ specification language.

A well-known typical example used in the literature of embedded hard real-time systems is the gas burner requirement which states that "gas must never leak for more than 4 time units in any period of at most 30 time units" [37]. In the language of scheduling this reads as: "a task must never use more than 4 time units of processing time in any period of at most 30 time units". Moreover, this is also the same as saying that a periodic resource model [31] with a period of 30 time units can only spend a budget of at most 4 time units. Here, a periodic resource model is a mechanism to ensure time isolation between different components, including tasks.

These properties cannot be specified using MTL without assuming that a bound over the input model exists. Although it is possible to construct arbitrary formulas in MTL emulating the counting of time for certain patterns (i.e., the ones we identify as being bounded) this shows to be unnatural due to the very large number of combinations required. RMTL-$\int$ shows to be intuitive and natural to express counting time as a first class citizen in the language and as a consequence is more succinct than MTL. Formulas that can relate elapsed time for two tasks such as "two tasks do not overload in a period of 30 time units if the (in-)equality $a = b - 10$ or $a < b + 10$ holds" can be expressed in RMTL-$\int$ but not in MTL, since a and b are variables that correspond to the execution time of each task, which are unbounded by definition. Let us now briefly review the syntax of the adopted specification language RMTL-$\int$.

a) RMTL-$\int$ formulae [9]: Let $\mathcal{P}$ be a set of propositions and $\mathcal{V}$ a set of logical variables. The syntax of RMTL-$\int$ terms η and formulas φ is defined inductively as follows:

$$\begin{aligned}\eta &::= \alpha \mid x \mid \eta_1 \circ \eta_2 \mid \textstyle\int^{\eta} \varphi \\ \varphi &::= true \mid p \mid \eta_1 < \eta_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \varphi_1 \ \mathrm{U}_{\sim\gamma}\ \varphi_2 \\ &\quad \mid \varphi_1 \ \mathrm{S}_{\sim\gamma}\ \varphi_2 \mid \exists x\, \varphi\end{aligned}$$

where: $\alpha \in \mathbb{R}$, $x \in \mathcal{V}$ is a logical variable, the operators $\circ \in \{+, \times\}$ are used for the sum and multiplication of terms, $\int^{\eta} \varphi$ is the duration of the formula φ in the interval $[0, \eta]$; $p \in \mathcal{P}$ is an atomic proposition, $<$ is the relation *less than* on terms, U and S are temporal operators, with $\sim \in \{<, =, \leq\}$ and $\gamma \in \mathbb{Q}_{\geq 0}$. RMTL-$\int$ formulas are evaluated over state sequences. A *timed state sequence* κ is an infinite sequence with finite variability of the form

$$(p_0, [i_0, {i'}_0[), (p_1, [i_1, {i'}_1[) \ldots,$$

where $p_j \in \mathcal{P}$, $i'_j = i_{j+1}$ and $i_j, i'_j \in \mathbb{R}_{\geq 0}$ such that $i_j < i'_j$ and $j \geq 0$. Let $\kappa(t)$ be defined as $\{p_j\}$ if there exists a tuple $(p_j, [i_j, {i'}_j[)$ such that $t \in [i_j, {i'}_j[$, and as $\emptyset$ otherwise. Note that there exists at most one such tuple and there are an $\epsilon_0 > 0$ and arbitrarily large n, m such that $|i_n - i_m| \geq \epsilon_0$ holds (divergence).

A *logical environment* is any function $\upsilon : \mathcal{V} \to \mathbb{R}_{\geq 0}$. For any $x \in \mathcal{V}$, $r \in \mathbb{R}$, and logical environment υ, we will denote

by $\upsilon[x \mapsto r]$ the logical environment that maps x to r and every other variable y to $\upsilon(y)$.

b) RMTL-$\int$ interval-based semantics: The truth value of a formula φ will be defined relative to a model (κ, υ, t) consisting of a timed state sequence κ, a logical environment υ, and a time instant $t \in \mathbb{R}_{\geq 0}$. We will write $(\kappa, \upsilon, t) \models \varphi$ when φ is interpreted as true in the model (κ, υ, t). Terms and formulas will be interpreted in a mutually recursive way.

For each formula φ, timed state sequence κ and logical environment υ, the auxiliary *indicator function* $1_{\varphi(\kappa,\upsilon)} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is defined as follows, making use of the satisfaction relation:

$$1_{\varphi(\kappa,\upsilon)}(t) = \begin{cases} 1 & \text{if } (\kappa, \upsilon, t) \models \varphi, \\ 0 & \text{otherwise.} \end{cases}$$

The value $\mathscr{T}[\![\eta]\!]_{(\kappa,\upsilon)\,t}$ of a term η relative to a model can then be defined. A *Riemann integral* [19] of the function $1_{\varphi(\kappa,\upsilon)}$ is used for the case of a duration $\int^{\eta} \varphi$.

$$\begin{aligned}
\mathscr{T}[\![\alpha]\!]_{(\kappa,\upsilon)\,t} &= \alpha \\
\mathscr{T}[\![x]\!]_{(\kappa,\upsilon)\,t} &= \upsilon(x) \\
\mathscr{T}[\![\eta_1 \circ \eta_2]\!]_{(\kappa,\upsilon)\,t} &= \mathscr{T}[\![\eta_1]\!]_{(\kappa,\upsilon)\,t} \circ \mathscr{T}[\![\eta_2]\!]_{(\kappa,\upsilon)\,t} \\
\mathscr{T}\left[\!\!\left[\int^{\eta} \varphi\right]\!\!\right]_{(\kappa,\upsilon)\,t} &= \begin{cases} \int_t^{t+t'} 1_{\varphi(\kappa,\upsilon)}(t_*)\ dt_* & \text{if } t' \geq 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

where t' is replaced by the term $\mathscr{T}[\![\eta]\!](\kappa, \upsilon)t$. The satisfaction relation is defined inductively as follows:

$$\begin{aligned}
&(\kappa, \upsilon, t) \models p && \text{iff} \quad p \in \kappa(t) \\
&(\kappa, \upsilon, t) \models \eta_1 < \eta_2 && \text{iff} \quad \mathscr{T}[\![\eta_1]\!]_{(\kappa,\upsilon)\,t} < \mathscr{T}[\![\eta_2]\!]_{(\kappa,\upsilon)\,t} \\
&(\kappa, \upsilon, t) \models \varphi_1 \vee \varphi_2 && \text{iff} \quad (\kappa, \upsilon, t) \models \varphi_1 \text{ or } (\kappa, \upsilon, t) \models \varphi_2 \\
&(\kappa, \upsilon, t) \models \neg\varphi && \text{iff} \quad (\kappa, \upsilon, t) \not\models \varphi \\
&(\kappa, \upsilon, t) \models \varphi_1\ \mathrm{U}_{\sim\gamma}\ \varphi_2 && \text{iff} \\
&\qquad \text{there exists } t' \text{ s.t. } t < t' \sim t + \gamma \text{ and } (\kappa, \upsilon, t') \models \varphi_2, \text{ and} \\
&\qquad \text{for all } t'' \text{s.t. } t < t'' < t', \ (\kappa, \upsilon, t'') \models \varphi_1 \\
&(\kappa, \upsilon, t) \models \varphi_1\ \mathrm{S}_{\sim\gamma}\ \varphi_2 && \text{iff} \\
&\qquad \text{there exists } t' \text{ s.t. } t - \gamma \sim t' < t \text{ and } (\kappa, \upsilon, t') \models \varphi_2, \text{ and} \\
&\qquad \text{for all } t'' \text{s.t. } t' < t'' < t, \ (\kappa, \upsilon, t'') \models \varphi_1 \\
&(\kappa, \upsilon, t) \models \exists x\, \varphi && \text{iff} \\
&\qquad \text{there exists a value } r \in \mathbb{R} \text{ s.t. } (\kappa, \upsilon[x \mapsto r], t) \models \varphi
\end{aligned}$$

We will write $(\kappa, \upsilon) \models \varphi$ as shorthand for $(\kappa, \upsilon, 0) \models \varphi$. Note that the semantics of the until operator is strict and non-matching. This implies that, in order to satisfy $\varphi_1\ \mathrm{U}_{\sim\gamma}\ \varphi_2$, the model is not required to satisfy φ_1. We will use the following abbreviations: $\varphi \wedge \psi$ for $\neg(\neg\varphi \vee \neg\psi)$, $\varphi \to \psi$ for $\neg\varphi \vee \psi$, $\Diamond_{\sim\gamma}\ \varphi$ for $true\ U_{\sim\gamma}\ \varphi$, and $\Box_{\sim\gamma}\ \varphi$ for $\neg(true\ U_{\sim\gamma}\ \neg\varphi)$.

An important property of our restriction is that RMTL-$\int$ satisfies by construction the Dirichlet condition implying the Riemann property [26, p.7]:

Lemma 1. *For any RMTL-$\int$ formula φ, timed state sequence κ, and logical environment υ, the indicator function $1_{\varphi(\kappa,\upsilon)}$ is Riemann integrable.*

This property ensures that many infinite discontinuities in the composition of formulas or terms cannot be achieved and validates the integrability of formulas over time. Infinite discontinuities can only come from the sequence definition that is somehow restricted to avoid such cases i.e. we say that the sequences are divergent.

B. λ-calculus.

The λ-calculus can be seen as an elegant computation model to synthesize temporal logic for different purposes such as runtime verification and/or shallow embedding in (automated) theorem provers. A λ-term is either a variable $x \in Var$, where Var is a countably infinite set of variables; an application of a function e_0 applied to an argument e_1, usually written $e_0\ e_1$; or a lambda abstraction, $\lambda x.e$ representing a function with input parameter x and body e. Formally, lambda expressions are inductively defined by

$$e ::= x \mid \lambda\, x.e \mid e_0\, e_1$$

where the metavariables e, e_0, e_1 represents a λ-calculus term.

An expression can be surrounded with parenthesis for clarity, and we use the notation with "."s to avoid the proliferation of multiple lambdas, each one with one argument. For instance, $\lambda x_1, \ldots x_n.M$ is equivalent to $(\lambda x_1(\ldots(\lambda x_n M)\ldots))$, where M is the body of the abstraction. We assume that lambda abstractions associate to the right, and applications to the left, i.e., $MN_1 \ldots N_n$ is equivalent to $(\ldots(MN_1)\ldots N_n)$. Note that λ acts as a variable binder in a similar way to the quantifiers $\exists$ and $\forall$ in *predicate calculus* and $\int \ldots dx$ in *integral calculus*.

We begin by describing the meaning of the β-reduction ($\longrightarrow_\beta$)

$$(\lambda x.M)N \longrightarrow_\beta M[N/x],$$

where $M[N/x]$ can be read "replace free occurrences of x in M by N". When applying the β-reduction rule it is often useful to also use the α-conversion rule, defined by:

$$\lambda x.M = \lambda y.M[y/x] \text{ and } y \text{ is not a free variable of } M.$$

This rule captures the fact that a bound variable can be replaced by any other fresh variable. The reduction denoted by $\longrightarrow_\beta^*$ is the transitive and reflexive clousure of $\longrightarrow_\beta$. Substitution suffers from the problem of "variable capture". It can be solved using different approaches. A simple one is to replace the bounded variables in certain circumstances as in [27], [20]. For instance, to evaluate $\lambda y.(\lambda x.yx)(x)$, we have that $(\lambda x.yx)[y/x]$. Here, using the modern approach, we need to use the α-reduction to rename x and reduce $(\lambda w.yw)[y/x]$ into $\lambda w.xw$.

III. Synthesis of RMTL-$\int$ Formulae

The synthesis algorithm for RMTL-$\int$ presented here is suitable to solve the satisfiability problem of our fragment using a variant of *cylindrical algebraic decomposition* (CAD). This means that our formalization is adjusted as an input model for SMT solvers in SMT-LIBv2 specification language [4], using a logic that supports *non-quantified linear arithmetic*, *uninterpreted functions with equality*, and *arrays*.

SMT provers have been progressively adding smart tactics for solving problems that until now could only be solved using human creativity. Of course several issues such as inductive proofs and quantified fragments are really difficult or even impossible to check by such general approaches. Due to being the target of several optimizations, such as conflict-driven

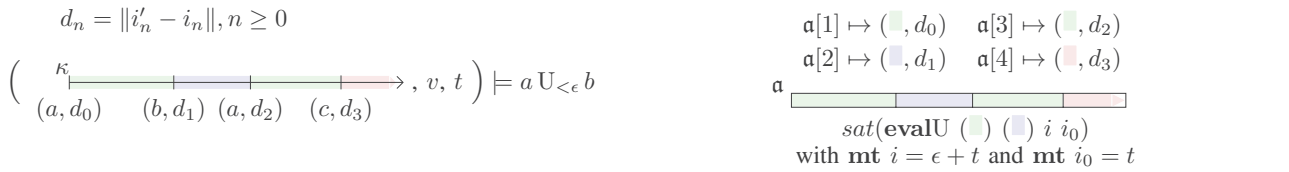

Figure 1. Timed sequence checking $a \,\mathrm{U}_{<\epsilon}\, b$ and equisatisfiable mapping

clause learning, and also due to their efficiency handling a mix of non-quantified logic fragments, including non-interpreted functions and decidable logic fragments for arithmetic, these solvers are suited for several classic problems in the real-time community. This fact has not been suitably explored until now; we give here the initial steps in this direction.

Efficient synthesis algorithms can give modular advantages for different problem formulations such as schedulability analysis. We will now describe a new algorithm for the synthesis of RMTL-$\int$ in which we encode the interval-based semantics of RMTL-$\int$ using lambda expressions, that will be converted further to the SMT-LIBv2 language with small effort. The set of constant arrays from propositions $\mathcal{P}$ to Boolean $\mathbb{B}$ is denoted by $\mathfrak{A}$. For arrays, we use the $select$ keyword that given a mapping of a timed state sequence $\mathfrak{a} \in \mathfrak{A}$ and an index $i \in \mathbb{N}_0$ returns a proposition. Any proposition p is in $\{\mathtt{tt}, \mathtt{ff}\} \subseteq \mathbb{B}$ and $\mathbf{ite}$ is the if-then-else construct.

In what follows we define the combinators **eval**p, **eval**U, **eval**$\int$, that will evaluate respectively propositions, less-until operators, and duration terms, based on the standard rewriting semantics of λ-expressions (i.e., the β-reduction). The other operators available in RMTL-$\int$ are directly translated. These include the traditional $\neg$ and $\vee$ operators and the arithmetic operators $+$ and $\times$. We introduce them for formulas as **eval**$<$, **eval**$\vee$, **eval**$\neg$, and **eval**$true$, and for terms as **eval**$+$, **eval**$\times$, **eval**x, and **eval**α. Constants and logical variables are locally substituted. The proposition formulation is encoded by the lambda expression $\mathbf{eval}p \doteq\ \lambda\ \mathfrak{a}\ p\ i\ .\ \mathbf{ite}\ (\mathbf{select}\ \mathfrak{a}\ i = p)\ \mathtt{tt}\ \mathtt{ff}$, where the $select$ keyword selects a given element of an array $\mathfrak{a}$ for some index i and returns a proposition. Arrays seem to be an intuitive way to encode sequences and anyone with experience in programming languages could follow the encoding, but it has its own drawbacks when considering non-linear polynomial arithmetic theory (Z3 handles it in isolation without offering a decision procedure when polynomials are combined with other theories).

We encode the timed state sequence as an array where each index identifies a state defined by a symbol (proposition) and a piece-wise constant interval (duration), meaning that time is piece-wise continuous. For that, we consider the function $\mathbf{mt} : \mathbb{N}_0 \to \mathbb{R}$ that given an index selects a cumulative duration, and the function $\mathbf{mt_v} : \mathbb{N}_0 \to \mathbb{R}$ that given an index returns the current duration. The Fig. 1 illustrates such mapping. $sat(\mathbf{eval}\mathrm{U}\ (\,)\ (\,)\ i\ i_0)$ corresponds to the satisfaction of the until operator's translation for two propositions and two indexes such that $i_0 < i$, $\mathbf{mt}\ i = \epsilon + t$ and $\mathbf{mt}\ i_0 = t$.

Lemma 2 (Sequence Equivalence)**.** *Let κ be a timed state sequence, $p \in \mathcal{P}$ a proposition, $\mathfrak{a} \in \mathfrak{A}$ an arbitrary constant array, and i an arbitrary index in $\mathbb{N}_0$. For all t and p, there exists an $\mathfrak{a}, i$ such that $p \in \kappa(t)$ is logically equivalent to* $\mathbf{select}\ \mathfrak{a}\ i = p$ *such that* $\mathbf{mt}\ i = t$.

Now, we continue with the translation of the formulas. The evaluation of the duration term denoted by **eval**$\int$ is defined by the set of lambda expressions, as follows:

$$
\begin{aligned}
ind \quad &\doteq \lambda \mathbf{eval}\ i\ .\ \mathbf{ite}\ (\mathbf{eval}\ i = \mathtt{tt})\ (\mathbf{mt_v}\ i)\ 0 \\
evale' \quad &\doteq \lambda f.\ \lambda \mathbf{eval}\ x\ i\ .\ (x \geq 0) \to \mathbf{ite}\ ((i \geq 0) \wedge (x > i)) \\
&\quad\ ((f\ f)\ \mathbf{eval}\ x\ i = ((f\ f)\ \mathbf{eval}\ (x-1)\ i) + (ind\ \mathbf{eval}\ x)) \\
&\quad\ ((f\ f)\ \mathbf{eval}\ x\ i = ind\ \mathbf{eval}\ x) \\
evale \quad &\doteq evale'\ evale' \\
\mathbf{eval}\textstyle\int \quad &\doteq \lambda i\ i'\ \mathbf{eval}\ .\ evale\ \mathbf{eval}\ (i-1)\ i'
\end{aligned}
$$

Given this definition, its inductive nature allows us to identify a pattern very close to the Riemann sum [22, p. 292], from which we derive the correction. The local variables i and i' are indexes from which t and t' are mapped such as $\mathbf{mt}\ i = t'$ and $\mathbf{mt}\ i' = t$. The evaluation of the duration term $\int^{\epsilon}$ (resp. $\mathbf{eval}\int\ i\ i'\ $) is complete if $\mathbf{mt}\ i = t + \epsilon$ and $\mathbf{mt}\ i' = t$ hold, for a given t. Note that the missing formula in the term is translated into an evaluation function and fed accordingly.

The evaluation of the less until $\mathbf{eval}\mathrm{U}$ is defined by the set of lambda expressions, as follows:

$$
\begin{aligned}
evali \quad &\doteq \lambda\ b1\ b2\ .\ \mathbf{ite}\ (b2 \neq \mathtt{ff})\ b2\ (\mathbf{ite}\ (b1 \neq \mathtt{tt})\ b1\ \mathfrak{r})), \\
evalb \quad &\doteq \lambda\ \mathbf{eval1}\ \mathbf{eval2}\ t\ v\ .\ \mathbf{ite}\ (v = \mathfrak{r}) \\
&\quad\ (evali\ (\mathbf{eval1}\ t)\ (\mathbf{eval2}\ t))\ v, \\
evalf' \quad &\doteq \lambda f.\ \lambda \mathbf{eval1}\ \mathbf{eval2}\ x\ i\ .\ (x \geq 0) \to \mathbf{ite}\ (i \geq 0 \wedge x > i) \\
&\quad\ (evalb\ \mathbf{eval1}\ \mathbf{eval2}\ x\ ((f\ f)\ (x-1)\ i) = (f\ f)\ x\ i) \\
&\quad\ (evalb\ \mathbf{eval1}\ \mathbf{eval2}\ x\ \mathfrak{r} = (f\ f)\ x\ i), \\
evalf \quad &\doteq evalf'\ evalf', \\
map2 \quad &\doteq \lambda x\ .\ \mathbf{ite}\ ((x = \mathtt{ff}_4) \vee (x = \mathfrak{r}))\ \mathtt{ff}\ \mathtt{tt}, \\
evalc \quad &\doteq \lambda \mathbf{eval1}\ \mathbf{eval2}\ i\ i'\ .\ evalf\ \mathbf{eval1}\ \mathbf{eval2}\ (i-1)\ i',\ and \\
\mathbf{eval}\mathrm{U} \quad &\doteq \lambda i\ i'\ \mathbf{eval1}\ \mathbf{eval2}\ .\ map2\ (evalc\ \mathbf{eval1}\ \mathbf{eval2}\ i\ i').
\end{aligned}
$$

Note that we need to remove the recurrence among the lambda expressions by unfolding until they converge. The following Example 1 illustrates this for a simple case of a duration evaluation.

Example 1. *The expression $evale$ 2 1 will be evaluated as follows:*

$$
\begin{aligned}
&evale\ \mathbf{eval}\ 2\ 1 \longrightarrow_\beta \\
&(\lambda x\ i.(x \geq 0) \to ite\ (i \geq 0 \wedge x > i) \\
&((evale'\ evale')\ \mathbf{eval}\ x\ i = ((evale'\ evale')\ \mathbf{eval}\ (x-1)\ i) + (ind\ \mathbf{eval}\ x)) \\
&((evale' evale')\ \mathbf{eval}\ x\ i = ind\ \mathbf{eval}\ x))\ 2\ 1 \longrightarrow^*_\beta \\
&(2 \geq 0 \to \mathbf{ite}\ (1 \geq 0 \wedge 2 > 1)\ (evale\ \mathbf{eval}\ 2\ 1 = (1 \geq 0 \to \mathbf{ite}\ (1 \geq 0 \wedge 1 > 1) \\
&(evale\ \mathbf{eval}\ 1\ 1 = (evale\ \mathbf{eval}\ 1\ 1 = ind\ \mathbf{eval}\ 1) + (ind\ \mathbf{eval}\ 1)) \\
&(evale\ \mathbf{eval}\ 1\ 1 = ind\ \mathbf{eval}\ 1)) + (ind\ \mathbf{eval}\ 2)) \\
&(evale\ \mathbf{eval}\ 2\ 1 = ind\ \mathbf{eval}\ 2))
\end{aligned}
$$

where after simplifying we get: $evale\ \mathbf{eval}\ 2\ 1 \longrightarrow^*_\beta (evale\ \mathbf{eval}\ 1\ 1 = ind\ \mathbf{eval}\ 1) + (ind\ \mathbf{eval}\ 2)$. *If we continue evaluating the expression we will always get* $evale\ 1\ 1 = ind\ \mathbf{eval}\ 1$ *and then we should stop evaluating the expression since we converge in* $(evale\ \mathbf{eval}\ 1\ 1 = ind\ \mathbf{eval}\ 1) = ind\ \mathbf{eval}\ 1$.

This encoding has this form to be easily translated into uninterpreted functions, since the evaluation always converges and all the possible assignments are defined. Interestingly, our definition can be adopted for monitoring (using functions interpreted as lambdas) and also to check satisfaction in a more general environment setting. For instance, given the uninterpreted function $evale : \mathbb{Z} \to \mathbb{Z} \to \mathbb{R}$, $evale$ is defined in AUFLIRA logic [4], as follows:

$$\forall x\ i, \quad \Big(\mathbf{implies}\ (x \geq 0)\ \Big(\mathbf{ite}\ ((i \geq 0) \wedge (x > i)) \\ (evale\ x\ i = (evale\ (x-1)\ i) + (ind'\ x)) \\ (evale\ x\ i = (ind'\ x))\Big)\Big)$$

Note that $\mathbf{eval}$ and $\mathbf{mt}$ are inlined in ind'. $\mathbf{mt}$ encodes the position at which the sequence shall be summed up by the *Riemann sum* constructed over the sequence κ mapped to $\mathfrak{a}$. The correctness result is built on top of the correspondence between the Riemann integrable property of the RMTL-$\int$ described in Lemma 1 and the Riemann sum. If we have logical variables, they are encoded as variables in the SMT solver with the domain in $\mathbb{R}$, and logical constants as constants in $\mathbb{Q}$. The translation of the operators $\mathrm{U}_{=\gamma}$, $\mathrm{U}_{\leq\gamma}$ and $\mathrm{S}_{\sim\gamma}$ are omitted here for simplicity since they are defined using the same pattern as the case of $\mathrm{U}_{<\gamma}$ resp. $\mathbf{eval}\mathrm{U}\ i\ i'$ such that $\mathbf{mt}\ i = t + \gamma$ and $\mathbf{mt}\ i' = t$ hold, for a given $t \in \mathbb{R}_{\geq 0}$.

Let us denote by Φ the set of RMTL-$\int$ formulas, by $\mathfrak{T}$ the set of RMTL-$\int$ terms, and by Φ_{smt} the set of *AUFLIRA* logic formulas [4]. To replace the inductive nature of the formulas, we have to define the function $rmtld\_synth : \Phi \to (\mathfrak{A} \to \mathbb{N}_0 \to \Phi_{smt})$ that applies the substitution of all the $\mathbf{eval}$ symbols (resp. $\mathbf{eval}$, $\mathbf{eval1}$ and $\mathbf{eval2}$) according to the inductive structure of the formula. In the same way, we consider the function $rmtld\_synth\_term : \mathfrak{T} \to (\mathfrak{A} \to \mathbb{N}_0 \to \Phi_{smt})$ that performs similar replacement for terms. Both functions are mutually recursive. The definition of such functions is simple because they align with the corresponding pattern throughout the formula structure. Due to the verbose nature of the functions $rmtld\_synth$ and $rmtld\_synth\_term$, we decided to omit them entirely here.

For the sake of showing a sound and complete synthesis procedure, we will formulate the notion of correctness.

Lemma 3 (Correctness)**.** *Let* κ *be a timed state sequence,* $\mathfrak{a}$ *an array that translates* κ*, and* t *a time instant. For any* ϕ *in RMTL-*$\int$*,* $(\kappa, \upsilon, t) \models \phi$ *iff* $(rmtld\_synth\ \phi)\ \mathfrak{a}\ t$ *is satisfiable.*

Note that the decision procedure for RMTL-$\int$ logic using linear arithmetic is incomplete. Our evaluation shows that in practice several problems could be solved in such a way and when necessary encoding in non-linear arithmetic theory is the only option. The encoding for this case is less intuitive for someone with functional programming background and is not addressed here.

IV. Formalization of Periodic Resource Models

Scheduling is the act of doing a timetable showing the times or dates where the resources are assigned according to some scheduling policy. In real-time systems literature, the scheduling is frequently characterized by a scheduling model that consists of three elements: a resource model, a scheduling algorithm (e.g. fixed priorities; earliest deadline first), and a workload model (e.g. task set).

We will assume *task sets* $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, such that $n \in \mathbb{N}^+$ is the number of tasks $\tau_i = (p_i, e_i)$ where p_i and e_i are, respectively, the period and the *worst-case execution time* (WCET) of τ_i. Each task $\tau_i \in \mathcal{T}$ is implicitly periodic and has implicit deadline. A *periodic resource model* ω is a tuple $(\mathcal{T}, \pi, \theta, rm)$, where $\mathcal{T} \subseteq \Gamma$, π is the *replenishment period* (i.e the budget reset period), θ is the *server budget*, and rm is the rate monotonic scheduling algorithm [3]. The set of periodic resource models is denoted by $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$ for an arbitrary $m \in \mathbb{N}^+$. We denote the index i of a task by τ_i and the index j of a resource by ω_j, where $0 < i \leq n$ and $0 < j \leq m$ holds, respectively. The outputs of a resource model ω are *sequences of events*.

Let us now consider the alphabet of events $\mathcal{E}$. Each element can be of one of the following types: a *task release event* RE; a *task start event* ST; a *task sleep event* SL; a *task resume event* RS; a *task stop event* SO; a *resource budget release event* RN; or a *general purpose event identifier tuple* EV. We also consider that general purpose events are special since they include a certain event identifier.

The set of tasks with higher-priority than (and including) τ_i for ω_j is denoted by $\gamma^{\tau_i}_{\omega_j}$. We also use h as the hyper period, the least common multiple of the periods of all the periodic tasks. For events, we adopt the following notations: $\mathsf{EV}(\omega_j, \cdot)$ denotes the set of events that can be generated by the resource model ω_j; $\mathsf{EV}(\omega_j, \tau_i)$ denotes the set of events that can be generated by the task τ_i in the resource model ω_j; $evs^+(\omega_j, \tau_i)$ is defined by $evs(\omega_j, \tau_i) \vee \mathsf{SO}_{(\omega_j,\tau_i)} \vee \mathsf{EV}(\omega_j, \tau_i) \vee \mathsf{RE}_{(\omega_j,\tau_i)}$, with $evs(\omega_j, \tau_i)$ defined by $\mathsf{ST}_{(\omega_j,\tau_i)} \vee \mathsf{RS}_{(\omega_j,\tau_i)} \vee \mathsf{RN}_{(\omega_j)}$, which specifies all events that a task τ_i in the resource model ω_j can trigger; $evs^-(\omega_j, \tau_i)$ denotes the formula resulting from the removal of the $\mathsf{RE}_{(\omega_j,\tau_i)}$ and $\mathsf{SO}_{(\omega_j,\tau_i)}$ events from $evs^+(\omega_j, \tau_i)$; finally, $evs^*(\omega_j, \tau_i)$ denotes the formula resulting from the removal of the $\mathsf{ST}_{(\omega_j,\tau_i)}$ and $\mathsf{SO}_{(\omega_j,\tau_i)}$ events from $evs^+(\omega_j, \tau_i)$. The binary operator $\varphi_1 \ominus \varphi_2$, meaning *next implies*, is a shorthand for $\varphi_1 \to (\varphi_1\, \mathrm{U}_{<b}\, \varphi_2)$, where b is a rational value that means the maximum duration of each event. These shorthands allow us to reduce the verbosity of the next definitions.

A resource model ω_j captures the semantic nature of a periodic resource model with a specification containing properties such as the resource model budget supply, the schedulability policy, the task set durations and period, and other intrinsic settings for complete specification of the resource. The resource model budget supply is specified by the formula

$$\Box_{\leq h}\ \mathsf{RN}_{(\omega_j)} \ominus \left(\Diamond_{=\pi}\ \mathsf{RN}_{(\omega_j)}\right) \wedge \int^{\pi} \bigvee\nolimits_{\tau_i \in \mathcal{T}} evs^+(\omega_j, \tau_i) \leq \theta, \tag{1}$$

where ω_j is one resource model, π and θ are their renewal period and budget, and $\mathsf{RN}_{(\omega_j)}$ is the budget renewal event.

This formula states that for each occurrence of the event $\mathsf{RN}_{(\omega_j)}$ in the resource model ω_j, the duration of the other events until π time units does not surpass the budget θ per period π.

For the partial order of the task releases, as defined by the scheduler policy rm, we introduce the RMTL-$\int$ formula

$$\Box_{\leq h} \bigwedge_{\tau_i \in \mathcal{T}} \left(\mathsf{RE}_{(\omega_j,\tau_i)} \ominus \left(ev(\omega_j, \tau_i) \ \mathrm{U}_{\leq p_i} \ \mathsf{SO}_{(\omega_j,\tau_i)} \right) \right), \tag{2}$$

where

$$ev(\omega_j, \tau_i) \triangleq \left(\bigvee_{\tau_k \in \gamma_{\omega_j}^{(\tau_{i-1})}} evs^+(\omega_j, \tau_k) \right) \vee evs^-(\omega_j, \tau_i)$$

and $\gamma_{\omega_j}^{(\tau_{i-1})}$ denotes the set of higher-priority tasks, excluding events triggered by the task τ_i. This formula means that for every event $\mathsf{RE}_{(\omega_j,\tau_i)}$ there is always an event $\mathsf{SO}_{(\omega_j,\tau_i)}$, and that the events occuring before $\mathsf{SO}_{(\omega_j,\tau_i)}$ should be any event from τ_i's higher-priority tasks.

The duration of tasks allocated to one resource model is specified by the formula

$$\Box_{\leq h} \bigwedge_{\tau_i \in \mathcal{T}} \mathsf{RE}_{(\omega_j,\tau_i)} \ominus \int^{p_i} \bigvee_{\tau_k \in \gamma_{\omega_j}^{(\tau_i)}} evs^+(\omega_j, \tau_k) \leq e_i. \tag{3}$$

Notice that the $\leq$ operator should be changed to $\geq$ in order to specify the absolute WCET of the task set only in case of interest.

We also specify other properties such as the precedence of the event $\mathsf{SO}_{(\omega_j,\tau_i)}$ (*i.e.*, each event $\mathsf{ST}_{(\omega_j,\tau_i)}$ may be followed by an event $\mathsf{SO}_{(\omega_j,\tau_i)}$, but the event $\mathsf{SO}_{(\omega_j,\tau_i)}$ occurs since $\mathsf{ST}_{(\omega_j,\tau_i)}$ occurs). The precedence of the event $\mathsf{SO}_{(\omega_j,\tau_i)}$ is specified by the formula

$$\Box_{\leq h} \bigwedge_{\tau_i \in \mathcal{T}} \mathsf{SO}_{(\omega_j,\tau_i)} \ominus \left(es(\omega_j, \tau_i) \ \mathrm{S}_{\leq p_i} \ \mathsf{ST}_{(\omega_j,\tau_i)} \right), \tag{4}$$

where

$$es(\omega_j, \tau_i) \triangleq \left(\bigvee_{\tau_k \in \gamma_{\omega_j}^{(\tau_{i-1})}} evs^+(\omega_j, \tau_k) \right) \vee evs^*(\omega_j, \tau_i).$$

The complete encoding of the component is given by the conjunction of the formulas 1, 2, 3 and 4. For the remaining part of the paper, we define it by $\mathsf{PRM}(\omega_j)$, where ω_j is indexed according to certain workload parameters, allowing us to unroll the sub-formulas in the correct way. This partially concludes the formalization of the periodic resource model's behavior using RMTL-$\int$.

Nevertheless, for the sake of having a decision procedure for schedulability checking, we have to introduce the following proposition.

Proposition 1 (Schedulability)**.** *Let ω_1 be a resource model. The resource model ω_1 is schedulable if and only if there exists a timed sequence such that $\mathsf{PRM}(\omega_1)$ is satisfiable.*

Compared to classic scheduling analysis, it is clear that our approach is more flexible to extend different time constraints and scheduling policies. In fact, this is a constructive approach that avoids reformulating every step of the analysis from the beginning. Most of the available scheduling analyses do not have this feature, since they do not use constraint programming approaches such as the one presented in this paper. In addition, the notion of scheduler synthesis is not addressed either. Here, we have a schedulability test and a mechanism to generate schedulers.

V. Evaluation

Currently, in strict terms, it is not possible to devise a fair evaluation comparison for our approach since there are no available tools that consider duration terms in the way we consider in this work. In order to provide some insight about the feasibility of our technique, we have measured the times taken by the Z3 SMT solver to prove satisfiability of a set of specifications, as shown in Table I. We have considered different structures for the presented formulae. The goal is to show indicators of the feasibility of the approach on sets of formulae with heterogeneous structural schemes, as we would expect to occur in real-life, hard real-time applications. Note that b and b' are sampled in increments of 5 units starting at 5 up to 45, and black cells mean timeout (more than 150s). More complex examples can be seen in the tool's repository.

To demonstrate in practice the schedulability analysis of resource models, a synthetic workload is described. Consider as example the workload composed by one component $(60, 50)$, meaning that it executes at each hyper period of the three tasks with parameter pairs $(20, 9)$, $(15, 8)$ and $(10, 3)$, and sets $50/60$ time units available for executing. The first element of the tuple is the period and the second the WCET. In order to describe it succinctly, let us assume micro expressions defined by

$$tk ::= tsk_{id}^{(p,e)} \mid tk_0 \succ tk_1 \mid tk_0 \bowtie tk_1$$
$$exp ::= id\,[tk]_{(\omega,\pi)} \mid exp_1 \parallel exp_2.$$

Remember that p, e are the period and the WCET, and ω, π are the budget and the period of the resource model. The expression describing the example is

$$server_0 \left[\left(tsk_{ts1}^{(20,9)} \succ tsk_{ts2}^{(15,8)} \right) \bowtie tsk_{ts3}^{(10,3)} \right]_{(60,50)}, \tag{5}$$

where it specifies that $ts1$ executes with higher priority than $ts2$, and $ts3$ runs arbitrarily with $ts1$ and $ts2$. To be able to show a compact table describing the translation, we will only adopt three events per task: RE (task job release), RU (task job running; start + resume) and SO (task job end; sleep + stop); and the event RN for resource budget release. The Equation 5 is translated into a set of RMTL-$\int$ formulas as described in Table II.

To better understand the micro expressions, we have the Examples 2 and 3. Note that the way micro expressions are exemplified follows the presentation of the Section IV where periodic resource models have first been introduced.

Example 2. *Take as an example the expression $tsk_{ts1}^{(20,9)}$. We define the meaning of the expression $tsk_{ts1}^{(20,9)}$, making use of the always operator, by the formula*

$$\mathsf{RE}_{(\tau_1)} \wedge \Box_{<h} \mathsf{RE}_{(\tau_1)} \rightarrow \left(\Diamond_{=p} \mathsf{RE}_{(\tau_1)} \right) \ominus$$
$$\left(\mathsf{ST}_{(\tau_1)} \vee \mathsf{RS}_{(\tau_1)} \vee \mathsf{SL}_{(\tau_1)} \ U_{<p} \ \mathsf{SO}_{(\tau_1)} \right) \wedge$$
$$\int^p \mathsf{ST}_{(\tau_1)} \vee \mathsf{RS}_{(\tau_1)} \vee \mathsf{SL}_{(\tau_1)} \vee \mathsf{SO}_{(\tau_1)} < e,$$

where h is the upper bound or hyper period, equal to 60, and $p = 20$, $e = 9$ is the task period and WCET. The whole expression means the starting point of the execution of a task $ts1$ that was represented by τ_1.

Example 3. *Consider now the priority expression $tsk_{ts1}^{(20,9)} \succ tsk_{ts2}^{(15,8)}$, and the shorthands τ_1 for task $ts1$ and τ_2 for $ts2$.*

ID	Formula	Sat?	Performance (5 10 … 45)
(a)	$p \wedge \Box_{<b}\ p \rightarrow \Diamond_{=2}\ p$	✓	
(b)	$(p \vee q)\ \mathrm{U}_{<b}\ r$	✓	
(c)	$\int^{b} p < 3$	✓	
(d)	$(p\ \mathrm{U}_{<b}\ q) \wedge \int^{9} q < 2$	✓	
(e)	$(p\ \mathrm{U}_{<b}\ q) \wedge 10 < \int^{9} q$	unsat	
(f)	$\Diamond_{<b}\ p \wedge \Box_{<b'} \neg p$	unsat	
(g)	$\Box_{<b'}(a \vee b)\ \mathrm{U}_{<b}\ r$	✓	

Table I
HEAT MAPS FOR PERFORMANCE COMPARISON USING THE RMTLD3SYNTH TOOL FOR SYNTHESIZATION AND THE Z3 SOLVER FOR CHECKING SATISFIABILITY (< 1S, $= 100$S, AND > 150S)

We get Φ'' equal to

$$\begin{aligned}&\left(\mathsf{RE}_{(\tau_2)} \ominus \left(\mathsf{ST}_{(\tau_2)} \vee \mathsf{RS}_{(\tau_2)} \vee \mathsf{SL}_{(\tau_2)} \vee Fl(\phi''')\ \mathrm{U}_{<15}\ \mathsf{SO}_{(\tau_2)}\right)\right) \quad \wedge \\ &\left(\mathsf{RE}_{(\tau_2)} \ominus \int^{15} \mathsf{ST}_{(\tau_2)} \vee \mathsf{SL}_{(\tau_2)} \vee \mathsf{RS}_{(\tau_2)} \vee \mathsf{SO}_{(\tau_2)} = 8\right) \quad \wedge \Phi''',\end{aligned}$$

where Φ''' is equal to

$$\begin{aligned}&\left(\mathsf{RE}_{(\tau_1)} \ominus \left(\mathsf{ST}_{(\tau_1)} \vee \mathsf{RS}_{(\tau_1)} \vee \mathsf{SL}_{(\tau_1)}\ U_{<20}\ \mathsf{SO}_{(\tau_1)}\right)\right) \quad \wedge \\ &\left(\mathsf{RE}_{(\tau_1)} \ominus \int^{20} \mathsf{ST}_{(\tau_1)} \vee \mathsf{SL}_{(\tau_1)} \vee \mathsf{RS}_{(\tau_1)} \vee \mathsf{SO}_{(\tau_1)} = 9\right) \quad \wedge \Phi,\end{aligned}$$

and the filter function $Fl(\Phi''')$ returns the formula

$$\mathsf{RE}_{(\tau_1)} \vee \mathsf{ST}_{(\tau_1)} \vee \mathsf{RS}_{(\tau_1)} \vee \mathsf{SL}_{(\tau_1)} \vee \mathsf{SO}_{(\tau_1)}.$$

Thus, the final expression is

$$\begin{aligned}&\left(\mathsf{RE}_{(\tau_1)}\ \mathrm{U}_{<2}\ \mathsf{RE}_{(\tau_2)}\right) \quad \wedge \\ &\Box_{<h}\left(\mathsf{RE}_{(\tau_1)} \rightarrow \left(\Diamond_{=20}\ \mathsf{RE}_{(\tau_1)}\right)\right) \wedge \left(\mathsf{RE}_{(\tau_2)} \rightarrow \left(\Diamond_{=15}\ \mathsf{RE}_{(\tau_2)}\right)\right) \quad \wedge \Phi'',\end{aligned}$$

and Φ is assumed initially as true. Note that the filter symbol Fl just filter propositions according to the occurrence in certain tasks. The remaining operators can be seen in Table II.

As a final remark, we are making a counter-intuitive assignment on tasks $ts1$ and $ts2$, since the priority under rate monotonic is governed by the lower period tasks which have the highest priority. And so our approach does not just focus on the rate monotonic but also on any possible assignment of tasks. For the whole experiments, the utilized setup has been an Intel Core i5-8365U CPU @ 1.60GHz with 16 GB of RAM memory, and running Fedora 31 X86'64.

VI. Related work

Although this work has, to our knowledge, been the first one to use modern SMT solvers for schedulability analysis of periodic resource models, it is far from being the first work combining formal specifications with schedulability analysis. In comparison with [8], a preliminary analysis, this paper describes all the details, intuition and ideas that were not described in the short version along with a use case.

a) Schedulability analysis based on deductive reasoning: Using temporal logics we have the works [36], [35], [32]. However, some of the works involving temporal logics make use of very creative steps using pen-and-paper proofs as the case of [36]. These attempts for formalizing real-time theory are error-prone and so they clearly point out that succinct and elegant formalisms for schedulability analysis of real-time systems are required. A formalization in Coq [5] proof assistant of an interval temporal logic known as duration calculus has been proposed by [35]. Along the lines of the previous ideas, Prosa [7] gives initial steps towards the construction of a modular theory to reason about schedulability analysis in Coq. Other works proposed by Wilding [34] have successfully replicated the informal optimality proofs for earliest-deadline-first scheduling policy.

b) Schedulability analysis based on automata reachability analysis: An earlier work for schedulability analysis of preemptive and non-preemptive scheduling using model checking has been proposed by Ferman and colleagues [16]. They have encoded the schedulability analysis problem as reachability analysis over a decidable class of timed automata including subtraction. The reachability problem for task automata, which are an extended timed automata with asynchronous processes, is undecidable as shown by Krčál and Yi [25]. The undecidability results were reached by considering that the execution time of the tasks are intervals (i.e., nonconstant execution time) and the completion time of a task influences the new task releases (e.g., as in dependent task models [29]). Some years later Fersman and colleagues [15], [14] have concluded that the schedulability problem can be solved more efficiently over the assumption that tasks are periodic and execute over a fixed priority policy. Regarding the multi-core scheduling, Krčál and colleagues [24] describe that the schedulability checking for multi-core systems with preemption is in general undecidable using task automata. In general, although undecidable, there is a wide class of real-time systems that can be specified using the decidable fragment.

VII. Conclusion and Future Work

In this paper, we have described a synthesis procedure translating the satisfiability problem for RMTL-$\int$ into the SAT instance modulo the theory of arrays, uninterpreted functions with equality and non-linear real-arithmetic. Using appropriate SMT solvers, the latter can then be used to check the satisfiability of the underlying formula. Recall that RMTL-$\int$ is a dense real-time version of MTL extended by a duration operator allowing to obtain quantitative measure for the satisfiability of formulas.

Other optimization techniques for synthesis of RMTL-$\int$ into modern SMT problems may be worth exploring. An example is the extension of the synthesis algorithm for continuous-based semantics without assuming intervals, and the consequent repetition of unbounded symbols.

Moreover, we have described an alternative approach to schedulability analysis following a formal specification of the components of the scheduling hierarchy using micro expressions. Using a translation to RMTL-$\int$ and its transformation into the SMTLIBv2 language for which we have used the Z3 solver, we obtained valid schedules. Our plan in terms of future work is to improve on the developments done so far and on the kind of system we target, in order to understand how the proposal scales for systems with characteristics even closer to those that are used in industry. The theory of strings (word equations) could also be adopted to solve the multi-core scheduling problem, instead of the array theory. However, it remains to be seen whether this can be better to explore interleaving of symbols. Finally, in the context of classic schedulability analysis, hybrid approaches can be adopted to treat global scheduling for multi-core systems.

$$prop_{fm} \triangleq \mathsf{RU}(c_0,\tau_1)\vee\mathsf{SO}(c_0,\tau_1)\vee\mathsf{RU}(c_0,\tau_2)\vee\mathsf{SO}(c_0,\tau_2)\vee\mathsf{RU}(c_0,\tau_3)\vee\mathsf{SO}(c_0,\tau_3)$$
$$init \triangleq \mathsf{RN}(c_0)\ \mathrm{U}_{<2}\ \big(\mathsf{RE}(c_0,\tau_1)\ \mathrm{U}_{<2}\ \big(\mathsf{RE}(c_0,\tau_2)\ \mathrm{U}_{<2}\ \mathsf{RE}(c_0,\tau_3)\big)\big)$$

$$\Box_{<60}\ \mathsf{RN}(c_0)\rightarrow\Big(\Diamond_{=60}\ \mathsf{RN}(\omega)\Big)\wedge\int^{60} prop_{fm}{<}50$$
$$\Box_{<60}\ \mathsf{RE}(c_0,\tau_1)\rightarrow\Big(\Diamond_{=20}\ \mathsf{RE}(c_0,\tau_1)\Big)\wedge\Big(\mathsf{RE}(c_0,\tau_1)\ \mathrm{U}_{<2}\ \big(\mathsf{RU}(c_0,\tau_1)\vee\mathsf{RU}(c_0,\tau_3)\vee\mathsf{SO}(c_0,\tau_3)\ \mathrm{U}_{\leq 20}\ \mathsf{SO}(c_0,\tau_1)\big)\Big)$$
$$\Box_{<60}\ \mathsf{RE}(c_0,\tau_2)\rightarrow\Big(\Diamond_{=15}\ \mathsf{RE}(c_0,\tau_2)\Big)\wedge\Big(\mathsf{RE}(c_0,\tau_2)\ \mathrm{U}_{<2}\ \big(\mathsf{RU}(c_0,\tau_2)\vee\mathsf{RU}(c_0,\tau_1)\vee\mathsf{SO}(c_0,\tau_1)\vee\mathsf{RU}(c_0,\tau_3)\vee\mathsf{SO}(c_0,\tau_3)\ \mathrm{U}_{\leq 15}\ \mathsf{SO}(c_0,\tau_2)\big)\Big)$$
$$\Box_{<60}\ \mathsf{RE}(c_0,\tau_3)\rightarrow\Big(\Diamond_{=10}\ \mathsf{RE}(c_0,\tau_3)\Big)\wedge\Big(\mathsf{RE}(c_0,\tau_3)\ \mathrm{U}_{<2}\ \big(\mathsf{RU}(c_0,\tau_3)\vee\mathsf{RU}(c_0,\tau_2)\vee\mathsf{RU}(c_0,\tau_1)\vee\mathsf{SO}(c_0,\tau_1)\vee\mathsf{SO}(c_0,\tau_2)\ \mathrm{U}_{\leq 10}\ \mathsf{SO}(c_0,\tau_3)\big)\Big)$$
$$\Box_{<60}\ \mathsf{RE}(c_0,\tau_1)\rightarrow\int^{20}\mathsf{RU}(c_0,\tau_1)\vee\mathsf{SO}(c_0,\tau_1){=}9$$
$$\Box_{<60}\ \mathsf{RE}(c_0,\tau_2)\rightarrow\int^{15}\mathsf{RU}(c_0,\tau_2)\vee\mathsf{SO}(c_0,\tau_2){=}8\ \wedge\ \Box_{<60}\ \mathsf{RE}(c_0,\tau_3)\rightarrow\int^{10}\mathsf{RU}(c_0,\tau_3)\vee\mathsf{SO}(c_0,\tau_3){=}3\ \wedge\ init$$

Table II
COMPLETE EXPANSION OF THE EQUATION 5 WHERE c_0 MEANS $server_0$

ACKNOWLEDGMENT

This work was partially supported by BMVI project IHATEC / SecurePort; by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UID/CEC/04234) and the INESC TEC (UIDB/50014/2020); also by the Norte Portugal Regional Operational Programme (NORTE 2020) under the Portugal 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) and also by national funds through the FCT, within project NORTE-01-0145-FEDER-028550 (REASSURE).

REFERENCES

[1] R. Alur and T. A. Henzinger. Back to the future: towards a theory of timed regular languages. SFCS '92, pages 177–186, Washington, DC, USA, 1992. IEEE Computer Society.

[2] R. Alur and T.A. Henzinger. Real-time logics. *Inf. Comput.*, 104(1):35–77, May 1993.

[3] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell, and A. J. Wellings. Fixed priority pre-emptive scheduling: an historical perspective. *Real-Time Syst.*, 8(2-3):173–198, March 1995.

[4] C. Barrett, P. Fontaine, and C. Tinelli. The SMT-LIB Standard: Version 2.5. Technical report, 2015. Available at `www.SMT-LIB.org`.

[5] Y. Bertot and P. Castran. *Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[6] P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of tptl and mtl. *Inf. Comput.*, 208(2):97–116, 2010.

[7] F. Cerqueira, F. Stutz, and B. B. Brandenburg. PROSA: A case for readable mechanized schedulability analysis. In *ECRTS 2016, Toulouse, France*, pages 273–284, 2016.

[8] A. De Matos Pedro, D. Pereira, L. M. Pinho, and J. S. Pinto. Smt-based schedulability analysis using rmtl-$\int$. *SIGBED Review*, 14(3):40–42, 2017.

[9] A. De Matos Pedro, J. S. Pinto, D. Pereira, and L. M. Pinho. Runtime verification of autopilot systems using a fragment of MTL-$\int$. *Int. J. Softw. Tools Technol. Transf.*, Aug 2018.

[10] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS'08/ETAPS'08*, pages 337–340, 2008.

[11] L. De Moura and N. Bjørner. Satisfiability modulo theories: Introduction and applications. *Commun. ACM*, 54(9):69–77, September 2011.

[12] B. Dutertre. Yices2.2. In *CAV'14*, pages 737–744, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[13] E. Allen Emerson. Handbook of theoretical computer science (vol. b). chapter Temporal and Modal Logic, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[14] E. Fersman, P. Krcal, P. Pettersson, and W. Yi. Task automata: Schedulability, decidability and undecidability. *Information and Computation*, 205(8):1149–1172, August 2007.

[15] E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. Schedulability analysis of fixed-priority systems using timed automata. *Theor. Comput. Sci.*, 354(2):301–317, March 2006.

[16] E. Fersman, P. Pettersson, and W. Yi. Timed automata with asynchronous processes: Schedulability and decidability. In *TACAS '02*, pages 67–82, London, UK, 2002. Springer-Verlag.

[17] C. J. Fidge. Real-time schedulability tests for preemptive multitasking. *Real-Time Syst.*, 14(1):61–93, January 1998.

[18] C.A. Furia and M. Rossi. On the expressiveness of MTL variants over dense time. FORMATS'07, pages 163–178, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] R.A. Gordon. *The Integrals of Lebesgue, Denjoy, Perron, and Henstock*. Graduate studies in mathematics. American Mathematical Soc., 1994.

[20] J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, New York, NY, USA, 2 edition, 2008.

[21] Y. Hirshfeld and A. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inf.*, 62(1):1–28, January 2004.

[22] D. Hughes-Hallett. *Calculus: Single Variable, 7e*. Wiley, 2017.

[23] D. Jovanović and L. de Moura. Solving non-linear arithmetic. In *IJCAR'12*, pages 339–354, Berlin, Heidelberg, 2012. Springer-Verlag.

[24] P. Krcal, M. Stigge, and W. Yi. Multi-processor schedulability analysis of preemptive real-time tasks with variable execution times. In *FORMATS'07*, pages 274–289, Berlin, Heidelberg, 2007. Springer-Verlag.

[25] P. Krčál and W. Yi. Decidable and undecidable problems in schedulability analysis using timed automata. In *TACAS'04, Barcelona, Spain.*, pages 236–250. Springer-Verlag, 2004.

[26] Y. Lakhneche and J. Hooman. Metric temporal logic with durations. *Theor. Comput. Sci.*, 138(1):169–199, February 1995.

[27] Christopher League. Lambda calculi: A guide for computer scientists by chris hankin. *SIGACT News*, 31(1):8–13, March 2000.

[28] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[29] W. Puffitsch, E. Noulard, and C. Pagetti. Off-line mapping of multi-rate dependent task sets to many-core platforms. *Real-Time Syst.*, 51(5):526–565, September 2015.

[30] P. Roux, M. Iguernlala, and S. Conchon. A non-linear arithmetic procedure for control-command software verification. In *TACAS'18*, pages 132–151. Springer International Publishing, 2018.

[31] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. RTSS '03, pages 2–, Washington, DC, USA, 2003. IEEE Computer Society.

[32] D. Shuzhen, X. Qiwen, and Z. Naijun. A formal proof of the rate monotonic scheduler. RTCSA '99, pages 500–, Washington, DC, USA, 1999. IEEE Computer Society.

[33] D. Souza and P. Prabhakar. On the expressiveness of mtl in the pointwise and continuous semantics. *Int. J. Softw. Tools Technol. Transf.*, 9(1):1–4, February 2007.

[34] M. Wilding. *A machine-checked proof of the optimality of a real-time scheduling policy*, pages 369–378. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[35] Q. Xu and N. Zhan. Formalising scheduling theories in duration calculus. *Nordic J. of Computing*, 14(3):173–201, September 2008.

[36] N. Zhang, Z. Duan, C. Tian, and D. Du. A formal proof of the deadline driven scheduler in PPTL axiomatic system. *Theor. Comput. Sci.*, 554(C):229–253, October 2014.

[37] C. Zhou and M. R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. Springer Publishing Company, 2010.